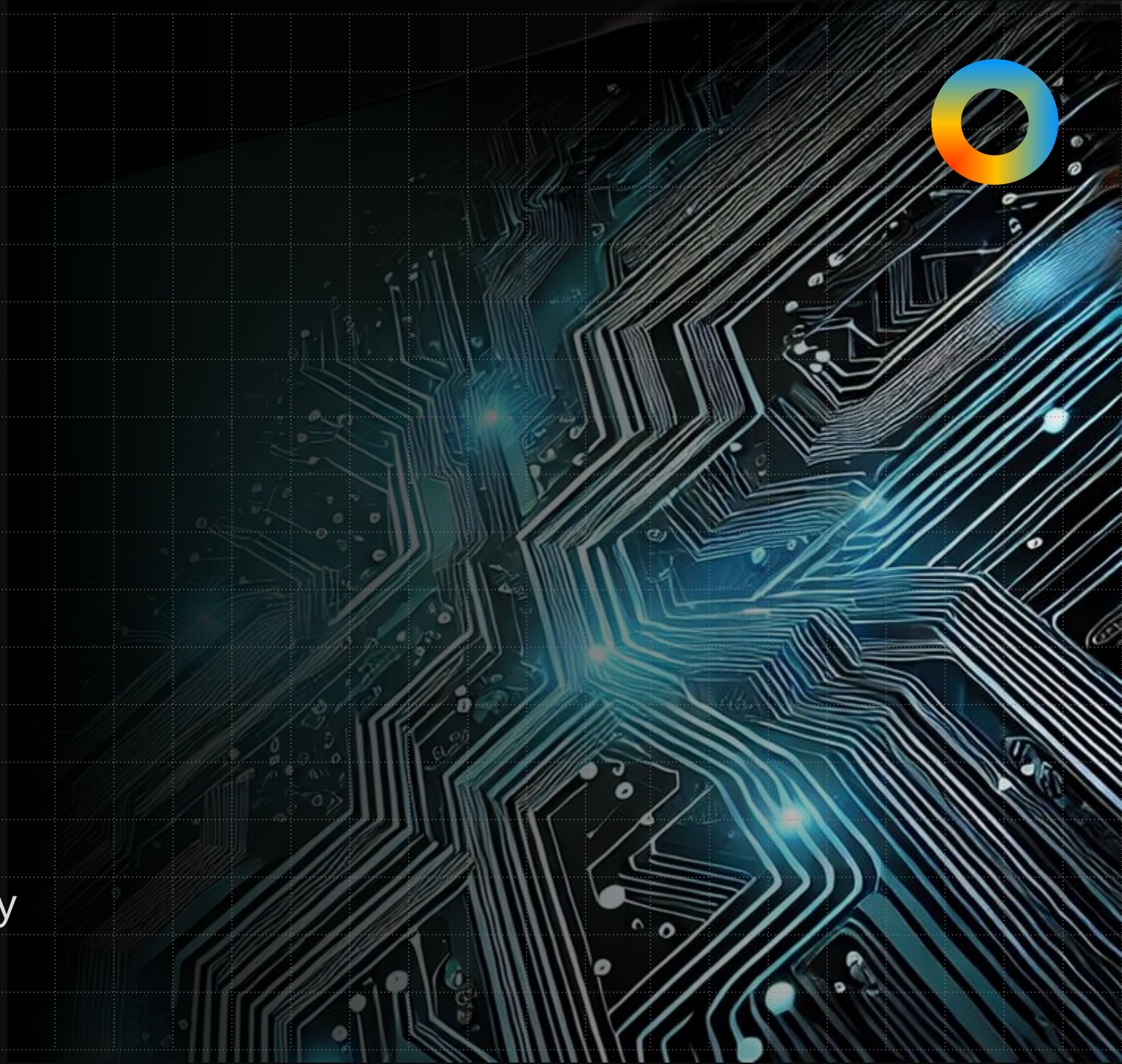
Python Programming

Collections

Dr. Chun-Hsiang Chan
Department of Geography
National Taiwan Normal University



Outlines

- List
- Tuple
- Set
- Dictionary
- Assignments

Before starting to know, …

- There are **four collection** data types in the Python programming:
- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
- **Dictionary** is a collection which is ordered** and changeable. No duplicate members.

https://www.w3schools.com/python/python_tuples.asp

Lists – Fundamentals

- List is the most powerful data type in Python, which I think at least. Because you may add or insert any data type into the list whereever you like. Usually, we can use the list as an array.

```
A = [1.2, 3.14, 100]
print(A)
print(type(A))
print(len(A))
B = [(1.2, 3.14, 100)]
print(B)
```

Lists – Indexing

- After knowing the list, there is onething that you have to know…

```
abc = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(abc[1])
print(abc[-1])
print(abc[1:])
print(abc[-5:])
print(abc[-3:-1])
```

Lists – Change

- After knowing the list, there is onething that you have to know…

```
abc = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10]
abc[1:4] = [2, 2, 2]
print(abc)

abc[5:] = [2, 2, 2]
print(abc)

abc.insert(3, 999)
print(abc)
```

Lists – Add

- After knowing the list, there is onething that you have to know…

```
# continue using the previous list for the following practice
abc.append(9999)
print(abc)

abc.extend([3333])
print(abc)

abc.extend((3333, 5555, 6666))
print(abc)
```

Lists – Remove

- After knowing the list, there is onething that you have to know…

```
# continue using the previous list for the following practice
abc. remove(9999)
print(abc)
abc.remove(10)
print(abc)
abc.pop(1)
print(abc)
```

```
abc. pop()
print(abc)

del abc[10]
print(abc)

abc.clear()
print(abc)
```

Lists – Sort

- Usually, you may want to re-order your dataset in some order.

```
# given two types of lists for list sorting
num = [3, 24, 13, 41, -50, 26, -17, 18, 99, 140, 1110, 190]
mystr = ['doctor','part','unique','college','taiwan','apple']
num.sort()
mystr.sort()
print(num)
print(mystr)
mystr.sort(reverse = True) # plz try ➔ mystr.reverse()
print(mystr)
```

Lists – Copy

- In data analysis, you may copy your list twice or more for different scenarios. **Notice:** you cannot just use b_list = a_list because b_list will only be a *reference* to a_list, and all changes you made on/in a_list will automatically also be made in b_list.

```
# make an experiment to prove it!
a_list = [1,2,3,4,5]
b_list = a_list
b_list[2] = 999
print(a_list) # what is the answer?
```

Lists – Copy

- So, how to copy a list?

```
# directly use the function of "copy"
a_list = [1,2,3,4,5]
b_list = a_list.copy()
b_list[2] = 999
print(a_list, b_list) # what is the answer?
# another method
b_list = list(a_list)
b_list[2] = 999
print(a_list, b_list) # what is the answer?
```

Lists – Join

- The last part is "join" – combining two or more list together.

```
# let mystr join into num
num = [3, 24, 13, 41, -50, 26, -17]
mystr = ['doctor','part','unique']
ns1 = num + mystr
print(ns1)
num.append(mystr)
print(num)
num.extend(mystr)
print(num)
```

Lists – Methods

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified value
pop()	removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

https://www.w3schools.com/python/python_lists_methods.asp

Tuples – Fundamentals

- Tuple is a very special data type in Python.
- To be honest, using tuple should consider twice because it is equipped the following characteristics:
  1) Ordered
  2) Unchangeable
  3) Allow duplicates

```
# make an experiment to prove it!
mytuple = (3, 24, 13, 41, -50, 26, -17, -50, 26, -17)
print(mytuple, mytuple[1])
mytuple[1] = 999 # can it work?
print(len(mytuple))
```

Tuples – Multi-type Tuples

- Some functionality in Tuple is just the same as that in List.

```
# different data types of tuples
tuple1 = ("apple", "banana", "cherry", "melon")
tuple2 = (1, 5, 7, 9, 3, 9, 3)
tuple3 = (True, False, False, True, False, True)
print(tuple1)
print(tuple2)
print(tuple3)
# multi-datatype tuples
tuple4 = ("abc", 56, 314, True, True, False, 40, "male")
```

Tuples – Indexing

- Indexing tuples …

```
# different data types of tuples
tuple1 = (1, 5, 7, 9, 3, 9, 3)
print(tuple1[2:])
print(tuple1[2:5])
print(tuple1[-2])
```

Tuples – Update

```
# add an element into the tuple
tuple1 = (1, 5, 7, 9, 3, 9, 3)
tuple2 = list(tuple1)
tuple2.append(1000)
print(tuple(tuple2))
# why do we need to
# transform into list at first?
X = ("apple", )
tuple1 += X
print(tuple1)
```

```
# remove an element from the
# tuple
Y = list(tuple1)
Y.remove("apple")
Y = tuple(Y)
```

Tuples – Unpack

- Due to the unchangeable nature of tuple, unpacking a tuple is very important.

```
# assign each tuple element for
# one variable
year1 = (1, 5, 7)
(joy, may, jon) = year1
print(joy)
print(may)
print(jon)
# we can also use asterisk (*) for unpacking
# the tuple; here, you need to observe the
# results of two examples and explain …
```

```
# example 1
year2 = (1, 5, 7, 9, 3, 9, 3)
(joy, may, *jon) = year2
print(joy)
print(may)
print(jon)
# example 2
(joy, *may, jon) = year2
print(joy)
print(may)
print(jon)
```

Tuples – Join Two or More Tuples & Methods

- As with other data types, a tuple also offers an ability of the joint.

```
# join tuples – by using addition
year1 = (1, 5, 7)
year2 = (12, 52, 72)
print(year1 + year2)
# join tuples – by using multiplication
year3 = year1*2
print(year3)
```

Tuples – Join Two or More Tuples & Methods

- **Tuple methods**

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Source: https://www.w3schools.com/python/python_tuples_methods.asp

Sets – Fundamentals

- A **set** is a collection that is ***unordered***, ***unchangeable****, and ***unindexed***.
- **Set items:** are unordered, unchangeable, and do not allow duplicate values.
- **Unordered:** means that the items in a set do not have a defined order. Set items can appear in a different order every time you use them and cannot be referred to by index or key.
- **Unchangeable:** Set items are unchangeable, meaning that we cannot change the items after the set has been created.

https://www.w3schools.com/python/python_sets.asp

Sets – Duplicated Values

- Due to the nature of set in Python, all elements in a set should be unique. Let's do an experiment.

```
# duplicated problem in a set
subject = {'math', 'english', 'sociology', 'math', 'physics'}
print(subject)
# True or 1 and False or 0
txtset = {3.5, 1, 0, 'math', False, True}
print(txtset)
# what do you observe in the second example?
print(len(txtset))
```

Sets – Add

- We can add an element, a set, or a list into a set.

```
# add an element into the set by using addition
subject = {'math', 'english', 'sociology', 'math', 'physics'}
subject.add('russian')
print(subject)
# add a set into the set by using update
subject2 = {'chinese', 'korean'}
subject.update(subject2)
print(subject)
# add a list into the set by using update
subject2 = ['chinese', 'korean']
subject.update(subject2)
print(subject)
```

Sets – Remove

- If you want to remove an element from the set, then …

```
# remove an element from the set by using remove
subject = {'math', 'english', 'sociology', 'math', 'physics'}
subject.remove('russian')
print(subject)
# remove an element from the set by using discard
subject.discard('math')
print(subject)
# delete all elements from the set
subject.clear()
print(subject)
```

Sets – Join1

- Combine two or more sets together, you may …

```
# join an element from the set by using union
subject = {'math', 'english', 'sociology', 'math', 'physics'}
subject2 = {'chinese', 'korean'}
subject.union(subject2)
print(subject)
# join an element from the set by using update
subject.update(subject2)
print(subject)
```

Sets – Join2 (Keep ONLY the Duplicates)

- Combine two or more sets together, you may …

```
# union - keep only the items that are present in both sets
subject = {'math', 'english', 'sociology', 'math', 'physics'}
subject2 = {'sociology', 'math', 'chinese', 'korean'}
subject.intersection_update(subject2)
print(subject)
# merging two sets by using intersection
subject.intersection(subject2)
print(subject)
```

Sets – Join3 (But NOT the Duplicates)

- Combine two or more sets together, you may …

```
# union - keep only the items that are present in both sets
subject = {'math', 'english', 'sociology', 'math', 'physics'}
subject2 = {'sociology', 'math', 'chinese', 'korean'}
# keep only the elements that are NOT present in both sets
subject. symmetric_difference_update(subject2)
print(subject)
# contains only the elements that are NOT present in both sets
subject. symmetric_difference(subject2)
print(subject)
# try the following test
x = {1, True}
print(subject. symmetric_difference(x))
```

Set Methods

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not

Source: https://www.w3schools.com/python/python_sets_methods.asp

Set Methods

Method	Description
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the summetric differences of two sets
symmetric_difference_update()	Inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Source: https://www.w3schools.com/python/python_sets_methods.asp

Dictionaries – Fundamentals

- Dictionary is also a powerful data type in Python; especially, one of the most common package, Pandas (or GeoPandas), has a useful class - dataframe, developed on the basis of dict.

```
# declare a dict
airport = {'air_name': 'TPE', 'Pax': 100}
print(airport)
print(airport['air_name'])
# duplicates are not allowed in dicts
airport2 = {'air_name': 'TPE', 'Pax': 100, 'Pax': 200}
print(airport2, len(airport)) # what does the length mean here?
```

Dictionaries – Index

- After declaration, again, we need to know how get the data.

```
airport = {'air_name': 'TPE', 'Pax': 100}
# get info of one attribute
print(airport['air_name'])
print(airport.get('air_name'))
# get all keys, values, and items
print(airport.keys(), '\n',airport.values() , '\n',airport.items()))
# add a key
airport['year'] = 1981
print(airport.keys())
```

Dictionaries – Change

- If you want to change or update the values in the dict, then …

```
airport = {'air_name': 'TPE', 'Pax': 100}
# change value in a dict by using direct indexing
airport['air_name'] = 'LHR'
# test if it changed
print(airport['air_name'])
# change value in a dict by using update
airport.update({'air_name' : 'KHH'})
# test if it changed
print(airport['air_name'])
```

Dictionaries – Add

- If you want to add new items into a dict, then …

```
airport = {'air_name': 'TPE', 'Pax': 100}
# add value in a dict by using direct indexing
airport['year'] = 1981
# test if it added
print(airport)
# added value in a dict by using update
airport.update({'year' : 1981})
# test if it added
print(airport)
```

Dictionaries – Remove

- If you want to remove new items into a dict, then …

```
airport = {'air_name': 'TPE', 'Pax': 100, 'year': 1981}
# remove value in a dict with a key
airport.pop('year')
print(airport)
# remove value in a dict by using popitem
airport.popitem()
print(airport)
# remove value in a dict by using del (notice: re-declare dict again)
del airport['Pax']
print(airport) # you may try airport.clear()
```

Dictionaries – Methods

Method	Description
clear()	Removes all the elements from the dictonary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key vlaue pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does ot exist: inset the key with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Source: https://www.w3schools.com/python/python_dictionaries_methods.asp

Assignment #01

- Given a name-score dictionary, please design a function to return the following statement:

  (1) return a set with all unique names

  (2) return the lowest and highest scores

  (3) return the corresponding name of the abovementioned scores

```
score_dict = {'Alice': 85, 'Bob': 90, 'Charlie': 78, 'David': 92, 'Eve': 85}
```

```
{'Alice', 'Bob', 'Charlie', 'David', 'Eve'}, (78, 92), ['Charlie', 'David']
```

Assignment #01

```
# Format
def report(score_dict):
    # annotation
    ...
    ...
    ...
    return course_set, score_tuple, correspond_name_list
```

The End

Thank you for your attention!

Email: chchan@ntnu.edu.tw

Website: https://toodou.github.io/